

Clashes between Culture and Software Development Methods: The Case of the Israeli Hi-Tech Industry and Extreme Programming

Orit Hazzan

*Department of Education in Technology & Science
Technion – Israel Institute of Technology
oritha@tx.technion.ac.il*

Yael Dubinsky

*Department of Computer Science
Technion – Israel Institute of Technology
yael@cs.technion.ac.il*

Abstract

This paper discusses connections between a national culture and the culture inspired by software development methods (SDMs). Specifically, based on our research on cultural issues related to software development teams, we propose a model that can help predict whether a specific SDM fits a specific national culture. This model first defines the terms 'tightness of an SDM' and 'tightness of a national culture'. With respect to an SDM, this term articulates the idea that the tighter an SDM is, the more ordered the software development process and environment it inspires will be; with respect to a national culture, the term 'tightness' reflects the degree to which a culture accepts and adopts ordered, planned and procedural work habits. The model then goes on to outline means that can help in the mapping of a given SDM, as well as a given national culture, along the following five dimensions: Project plan, procedures and standards, responsibility, time estimation, and need satisfaction. Finally, based on these mappings, the fitness of a given SDM and a national culture is examined. It is proposed that this fitness can serve as a tool for predicting the degree to which a given SDM will be accepted by a specific national culture in general, and by a specific team that is part of that culture, in particular. The model is illustrated using the Israeli hi-tech industry as an example of a national culture and Extreme Programming (XP) as an example of an SDM.

1. Introduction

Consider the two following data, referring to the year 2001:

- "Silicon Valley contains over 7,000 established high-tech firms and a significant number of start-ups in a given year".¹
- Israel's population is about 6 million. In 2001, there were about 3,000 start-up companies in Israel² – the second highest number of start-ups after the US (in absolute, not relative, terms).

If we compare Israel's population with the USA's population, these numbers seem quite unreasonable.

¹ Source:
http://sites.state.pa.us/PA_Exec/DCED/tech21/be-silicon.htm

² Source:
http://www.ite.poly.edu/htmls/role_israel0110.htm

Our purpose in this article is to present a model that explains these apparently contradicting facts. The model also explains the extraordinary success of the Israeli hi-tech industry in the dot.com era. This, however, is not the only objective of the proposed model. The model also serves as a means to estimate the degree to which a given software development method (SDM) will be accepted by a specific software development culture.

The model is based on the concept of *tightness*. Both an SDM and a culture are said to be tight if they inspire strict and ordered habits. More specifically, with respect to an SDM, the term tightness reflects the degree to which an SDM advocates an ordered software development process; with respect to a national culture, the term tightness reflects the degree to which work is performed according to given procedures that characterize this culture.

The tightness level of cultures and SDMs is ranked along five dimensions, each of which specifies a dimension related to software project management: project plan, procedures and standards, responsibility, time estimation, and need satisfaction.

Based on this model, we suggest an analysis of national cultures and SDMs and an examination of potential clashes when a gap exists between their tightness levels. It is suggested that the wider the gap between the tightness levels of these two entities, the harder it is to implement the SDM in that culture. We illustrate our model using the Israeli hi-tech software industry as an example of a culture, and Extreme Programming (XP) as an example of an SDM, and further discuss the fitness of XP to the culture of the Israeli software industry.

The connection between the two objectives of our model (i.e. to explain the success of the Israeli hi-tech industry and to estimate the degree to which a given SDM will be accepted by a specific software development culture) is explained by the fact that, on the one hand, Israel succeeded during the hi-tech peak due to its low tightness, while, on the other hand, it is the low tightness of the Israeli culture that at present resists the acceptance of tight SDMs. We emphasize that the tightness scale does not indicate neither any 'rightness' scale nor any 'quality' scale. It does aim at providing additional perspective at software development processes.

The motivation for this study arose from our observation, based on close interaction with many Israeli software developers and on the use of several research tools, that XP has not been embraced by the Israeli software industry. Indeed, the proposed model explains this phenomenon. According to our model, XP ranks high

on the tightness scale. This is mainly because XP outlines specific practices that are implemented on a daily basis. At the same time, our model suggests that the Israeli culture ranks low on the tightness scale. This is mainly expressed by Israeli developers' tendency to improvise during software development processes. Based on these two rankings we explain the potential clashes between the Israeli software industry and XP that can explain why XP is not adopted naturally by Israeli software teams.

Connections between SDMs and cultural issues have been discussed previously (Yourdon, 1997; Sawyer and Guinan, 1998; Abrahamsson *et al.*, 2002). We propose to expand the discussion on the connections between SDMs and culture-related issues on the national level. Furthermore, although organizational culture is a topic that is discussed among the XP community in general, and data on resistance to the adoption of XP is already available in particular (Auer and Miller, 2001A), we feel that in the case of the Israeli software industry we are witness to more than merely an organizational resistance, and that this evident resistance can be explained on the national level.

Our analysis suggests that this combination (the Israeli culture and XP) yields results that can increase awareness to problems that might arise due to clashes between an SDM and a *national* culture, within which an attempt is made to implement the SDM. Based on this exploration, we aim to present a partial answer to the question of whether a specific SDM fits a specific national culture.

Section 2 of the paper presents the theoretical background of our research, addressing culture-related issues connected to SDMs, the Israel hi-tech industry and XP. In Section 3 we present our model and in Section 4 we illustrate the model using the example of the Israeli software industry and XP. In Section 5 we discuss our model and suggest future research directions.

2. Theoretical Background

2.1. Culture in software organizations

As mentioned above, this paper deals with possible connections between cultural characteristics and the willingness of software teams to adopt a given SDM. The first question that comes to mind is: *why, in the first place, should such relations be explored?*³

In what follows, we will try to answer this question. As it is well known, software development entails many problems (Brooks, 1987; Mullet, 1999), including clashes in customer-developer communication, bugged software, difficulties in program comprehension, and mis-

understandings among team members. One of the most powerful pieces of data that illustrates these problems is the high percentage of software systems that do not fit customers' requirements, and which is estimated to be between 50 and 80% (Mullet, 1999).

The fact that so many problems exist in the software industry can be explained by the fact that software development is a contradiction-ridden process. For example, *creativity* is needed to find solutions in short time-to-market industries whereas *discipline* is needed for the production of high-quality software products (just think about the discipline required for testing!); or the contradiction between the *cooperation* needed among team members for the success of software projects as opposed to the *competition* exhibited by many software developers in this competitive environment. We propose that finding the right balance between such extremities is one of the secrets to successful software projects. This paper attempts to examine this balance from a national-cultural perspective.

As mentioned previously, culture in general, and organizational culture in particular, is a topic that is discussed in the literature with respect to SDMs. Usually, these discussions focus on the organizational level. For example, Highsmith (2002) says that "More methodology implementations flounder from failing to adequately account for an organization's cultural factors than for any other reason. A particular culture is not necessarily change tolerant or change resistant—but it may resist certain types of changes and embrace others. ... Many methodology failures are caused by a problem definition followed by a solution design, with little analysis of whether or not the solution design fits the company or the project team's culture."

Another theory, which focuses on the organizational culture, corresponds the software development paradigm (SDP) to the maturity of the developed product and the culture within which it is developed. According to Moore (2000), there are four basic organizational cultures: cultivation, competence, collaboration, and control, to which he matches one of three methodology categories: rigorous (RM), agile (AM), or ad hoc or no methodology (NM). A cultivation culture is motivated by self-realization and can be illustrated by Silicon Valley start-up companies, which fit the NM category. A competence culture is driven by the need for achievement; collaboration cultures are driven by a need for affiliation, and control cultures are motivated by the need for power and security. While the agile approach fits the competence and the collaboration cultures, the RM fits the control culture. Highsmith (2002) associates each methodology to a specific market phase. According to Highsmith, while the NM approach fits the initial phases of software product development, at later stages, when close interaction with customers is required, the AM approach fits better. During the Main Street market phase, the RM approach fits in the best.

This perspective is referred to towards the end of the next section, which describes the Israeli hi-tech industry.

³ We would like to emphasize that we are not addressing problems that may arise when developers from different cultures work together (as described, for example, in Walenta, 2004). Rather, we focus on problems that originate in gaps between a national culture and the culture that is inspired by a given SDM.

2.2. The Israeli hi-tech industry⁴

Israel is a very small country with a population of about 6.5 million people. Still, at its hi-tech economic peak, Israel was one of the world's leading centers for technology start-ups and innovations. As mentioned before, despite its small population, Israel had, at that time, about 3,000 start-ups⁵, and came in third (after the USA and Canada) on the list of countries with the highest number of companies listed on NASDAQ.

This blossoming is explained by two main factors. The first is the issue of national security and military needs that led to the development of cutting-edge technologies. Since its establishment in 1948, Israel has had to invest huge budgets and efforts to maintain its military advantage in order to survive. In particular, designated army units exist that specialize in technological innovations. As it turns out, many of Israeli's hi-tech entrepreneurs started their careers in the Israeli army.

The second factor that explains the success of the Israeli hi-tech industry in the 1990s is the massive immigration wave of Russian engineers from the former Soviet Union during the 1990-2000 decade. This addition of engineers to the Israeli population led Israel to have the highest number of engineers per capita in the world⁶.

The peak of this period was in the year 2000. Foreign investors poured billions of US dollars into Israeli start-ups and venture capital funds. In that year, hi-tech exports reached record highs and comprised 57% of the entire export from Israel. Many companies that were founded in Israel during that period moved their headquarters to the U.S., where the lion's share of their customer base was located, but often left their research and development operations in Israel.

Some of these companies were purchased by American companies. For example, Mirabilis (the inventor of ICQ - I Seek You) was purchased in June 1998 by AOL for \$287 million⁷; in May 2000, Chromatis (which developed optical networking) was purchased by Lucent Technologies for about \$4.5 billion (and was shut down by Lucent in August 2001)⁸.

The strength of the Israeli software firms was, and still is, in their innovative and daring R&D. Not surprisingly,

to this day, IBM, Intel and Microsoft operate research and development centers in Israel.

It seems that during the hi-tech peak, the Israeli hi-tech industry was in its early market phase and the cultivation culture was dominant. As was explained in Section 2.1, no methodology fits this combination. Indeed, this was the accepted situation in software development teams. As will be illustrated later on, this fact is reflected by the low tightness that characterizes the Israeli software culture even today. We propose that since the market conditions have changed in the past four years, and since the Israeli software industry is not a start-up based industry any more, the Israeli software industry should change its perception with respect to the need to work according to software development methods.

2.3 Extreme Programming

In this section we describe the unique features of XP and discuss the resistance that it sometimes arouses.

XP is one of the agile software development methods. It consists of the following four values: communication, feedback, courage, and simplicity. Twelve practices that define the XP development environment and process are derived from these values, as follows: test-driven development, refactoring, on-site customer, metaphor, pair programming, continuous integration, sustainable pace, simple design, planning game, coding standards, short releases, and collective ownership. Using XP means that the team implements all 12 XP practices (Beck, 2000).

As can be seen, the practices relate to all aspects of software development. The adjective Extreme means that XP applies 12 practices, which are all well known in the software industry, but takes them to their extreme in order to make the most of their contribution to the software development process. In addition, unlike many other software development methods that are described in terms of development stages or phases, XP defines work procedures and outlines daily work activities related to software project management, code development, time management and relationships with customers.

As it turns out, the agile approach in general, and XP in particular, raise some resistance⁹. Indeed, XP is a new paradigm in the software development world. In light of Kuhn's perspective on the acceptance of new paradigms in the scientific world (Kuhn, 1962), the fact that a new paradigm is not easily accepted by the relevant community comes as no surprise.

Following are three pieces of evidence of this resistance, as is reflected with respect to XP. The first is taken from an interview with Kent Beck, who said¹⁰: "On the technical side, a lot of programmers sort of like being

⁴ This description is based largely on Tomayko and Hazzan (2004).

⁵ Source:
http://www.ite.poly.edu/htmls/role_israel0110.htm.

⁶ Source:
http://www.smartcodecorp.com/about_us/israel_profile.asp.

⁷ Source:
<http://www.ccta.ca/english/publications/submissions/1999-98/1998-10-01.htm>

⁸ Source:
<http://www.lucent.com/press/0500/000531.coa.html>

⁹ See for example:
<http://christiansepulveda.com/blog/archives/000018.html>,
<http://homepage.mac.com/keithray/blog/2003/07/03/>

¹⁰ Look at:
<http://www.cutter.com/research/2002/edge020903.html>

able to go into their little caves and not have much accountability. They have a big problem with weekly planning, very visible estimation, programming with other people, testing and integration, and being a continuous part of the process. There may be some resistance there." The second is taken from Ken Auer and Roy Miller's book *Extreme Programming Applied: Playing to Win* (Auer and Miller, 2001B). They say that "XP forces people out of their comfort zones. They will resist. This resistance comes from fear and pride. Overcome this by focusing on using XP as a strategy to increase their chances of winning." The third is from Joshua Kerievsky, who describes his experience teaching XP as follows: "If you teach XP, you will encounter resistance. Experienced software people often have the strongest objections to XP before they come to understand it. Knowing how best to handle the various forms of resistance is essential to effectively teach XP."¹¹

3. A Five Dimensional Model: Relationship between Culture and Acceptance of an SDM

In this section, we present a model, according to which the suitability of an SDM to a particular national culture can be analyzed by measuring the tightness level of these two elements. According to our model, the tightness levels of an SDM and of a culture are measured along the following five dimensions:

- Project plan
- Procedures and standards
- Responsibility and accountability
- Time estimation
- Need satisfaction (Hazzan and Dubinsky, 2003).

We now specify for each component, i.e. the SDM and the culture, those factors that determine the tightness level with respect to each dimension. In both cases, unless specified otherwise, the higher the values of these factors, the tighter the SDM or the culture.

Tightness level of an SDM:

- Project plan dimension: number of releases and feedback milestones, level of emphasis put on planning, number of days for which *specific* planning is made.
- Procedures and standards dimension: level of detail that describes the SDM.
- Responsibility and accountability dimension: the frequently at which team members are required to report on their progress.
- Time estimation dimension: the importance given to time estimation in general, and the resolution level of the time estimation (hours, days, months) in particular (the smaller the time units, the higher

is the resolution, and the higher is the value of this dimension).

- Need satisfaction dimension: the mutual dependency of team members, level of planning that inspires the message of "invest now for the future".

Tightness level of a culture:

- Project plan dimension: the cultural tendency to plan ahead.
- Procedures and standards dimension: the cultural tendency to follow the plan (without improvising and taking shortcuts).
- Responsibility and accountability dimension: the tendency to avoid taking unnecessary risks.
- Time estimation dimension: the tendency and commitment to development time estimation and to retrospective evaluation of these estimations.
- Need satisfaction dimension: the level to which a culture imparts the importance of the investment of time in the present in order to gain long-term benefits, the degree to which people in the culture agree to maintain a degree of mutual dependency.

While an SDM can be analyzed theoretically using this model, we have developed a questionnaire for the analysis of the tightness level of a culture (see Appendix 1). The questionnaire was constructed based on data gathered using several research tools that enabled us to learn about software developers' perceptions with respect to the five aforementioned dimensions.

Table 1 shows four possible pairs of SDMs and cultures according to their tightness level. In the next section (Section 4), focus is placed on the Hard Implementation quarter in which an SDM with a high level of tightness is implemented in a culture characterized by low tightness. This situation is illustrated by the Israeli software teams' resistance to adopt XP. In this case, the Israeli culture is an example of a culture with a low level of tightness and XP is an example of an SDM with a high tightness level. As Table 1 suggests, in such a situation it may be difficult to implement the SDM in the culture. The other three cells of Table 1 are not labeled and can be the subject for another research.

Table 1: Tightness matrix of SDMs and cultures

		SDM Tightness	
		<i>Low</i>	<i>High</i>
Culture Tightness	<i>Low</i>		Hard Implementation
	<i>High</i>		

4. The Israeli Hi-Tech Industry and XP

This section first explains why the Israeli culture is ranked low on the tightness scale (Section 4.1) while XP is ranked high on the same scale (Section 4.2). Then, based on these explanations, we suggest possible clashes between the Israeli software culture and XP (Section 4.3).

¹¹ Source: Extreme Educational Symposium at the 4th International Conference, XP 2003, Genova, Italy, <http://www.xp2003.org/EduSymCfP.html>.

4.1. Tightness of the Israeli hi-tech culture

In this section we present data that indicates the low ranking on the tightness scale of the Israeli software culture. This analysis is preliminary and further data collection and analysis are on-going.

As it turns out, creativity and improvisation are accepted as the most characteristic features of the Israeli software industry. Accordingly, careful planning and working according to an ordered procedure are not a common occurrence in the Israeli software development culture. Unless referring to a software company in which a CMM (Capability Maturity Model) procedure is enforced, or an American company that maintains a site in Israel, our acquaintance with Israeli software development teams tells us that working according to a detailed and defined procedure is actually quite rare among Israeli software teams.

Following are two data that illustrate this observation.

Data 1: At the *IEEE International Conference on Software – Science, Technology & Engineering*, which was held in Herzelia, Israel in November 2003, we distributed the questionnaire presented in Appendix 2. One of the questions was:

If you are familiar with an Israeli software team, please indicate what main characteristic of the Israeli software industry led, in your opinion, to the success of the Israeli software industry in the 1990s?

Out of 27 forms completed, 15 participants answered this question. Several answers listed more than one characteristic. Table 2 presents the results. As can be seen, factors are mentioned that indicate low tightness along the project plan, procedures and standards and need satisfaction dimensions of our model.

Table 2: Some results of the questionnaire presented in Appendix 2

Factor	Number of answers
Creativity, originality, inventive skills	8
Improvisation	4
Intensive work, speed, agility (quickness)	4

Data 2: The questionnaire presented in Appendix 1 was responded to by 43 software engineering students, most of them worked in the Israeli hi-tech while studying. Table 3 presents several illustrative results.

As can be observed, the majority of the students prefer to work in small teams in order to foster software development processes and they appreciate the role of improvisation in such processes. Specifically, the two first items illustrate low tightness of the need satisfaction dimension. That is, in order to satisfy the individual's need to advance the development process, students tend to forego other factors that might be important in software development processes. The third item indicates low

tightness with respect to the project plan dimension: Since we spend such long hours at work and so much time is available, we have no need to plan our work. The fourth item clearly illustrates low tightness along the procedures and standards dimension.

Table 3: Some results of the questionnaire presented in Appendix 1

Statement	Agree	Tend to agree	Tend to disagree	Disagree	N/A
It is preferable to minimize, as much as possible, the dependency level among software team members.	23	11	6	1	2
It is preferable to work in small teams in order to foster decision-making processes as much as possible.	8	25	8	1	1
When I chose a profession, I took into consideration that I would have to devote many hours everyday at work and give up my personal life.	6	20	13	4	
Intuition and improvisation are important in software development processes.	17	21	4	1	

4.2. The Tightness of Extreme Programming

In this section we first rank the XP practices on a tightness scale of 1 to 4 (Table 4). Specifically, the higher the rank of a specific practice on the tightness scale, the more detailed the work habits and procedures outlined by the practice are the more precise is the actual manner in which the practice is applied in practice. As can be seen, most of the XP practices are either tight or very tight.

Table 4: XP practices according to tightness level

Tightness scale	XP practices
1 - loose	Metaphor
2 - reasonable	Planning game, Small releases, Simple design
3 - tight	Refactoring, Sustainable pace, Customer on-site
4 - very tight	Test-driven development, Continuous integration, Coding standards, Pair programming, Collective ownership

Second, in order to gain a more detailed analysis, we examined the contribution of the XP practices to the tightness level of XP along the five dimensions of our model (see Table 5). With respect to each XP practice outlined in Table 5, we mention only those dimensions whose tightness level is impacted significantly and directly by the practice.

Table 5: Contribution of the XP practices to the tightness level of XP along the five dimensions of the proposed model

Practice	Dimensions to whose tightness level the practice contributes
Test-driven development	<ul style="list-style-type: none"> • Procedures and standards • Responsibility and accountability • Need satisfaction
Refactoring	<ul style="list-style-type: none"> • Procedures and standards • Need satisfaction
On-site customer	<ul style="list-style-type: none"> • Project plan • Procedures and standards • Time estimation
Metaphor	<ul style="list-style-type: none"> • Need satisfaction
Pair programming	<ul style="list-style-type: none"> • Procedures and standards • Responsibility and accountability • Need satisfaction
Continuous integration	<ul style="list-style-type: none"> • Procedures and standards • Need satisfaction
Sustainable pace	<ul style="list-style-type: none"> • Procedures and standards • Time estimation • Need satisfaction
Simple design	<ul style="list-style-type: none"> • Procedures and standards • Need satisfaction
Planning game	<ul style="list-style-type: none"> • Project plan • Procedures and standards • Time estimation
Coding standards	<ul style="list-style-type: none"> • Procedures and standards • Need satisfaction
Short releases	<ul style="list-style-type: none"> • Project plan
Collective ownership	<ul style="list-style-type: none"> • Need satisfaction

4.3. XP and the Israeli culture

Based on Sections 4.1 and 4.2, this section examines the fitness of XP to the culture of the Israeli software industry.

In the introduction we explained our choice of the pair XP–Israeli Culture. We will now elaborate some more on our choice of this pair. During the past two years, in which we maintained close interaction with many software development teams who invited us to present XP to them, we received ambiguous messages. On the one hand, people express a lot of interest in hearing about XP, how it can improve their software development processes, how it can help them solve basic problems that they encounter on a daily basis, etc. On the other hand, we feel a strong resistance towards XP, expressed at different levels of the organizational hierarchy. This observation led us to develop the above-

described model based on which we will now attempt to explain this phenomenon.

Specifically, we will first explain how XP *values* fit the tightness level of the Israeli culture in general, and the culture of Israeli software organizations in particular (hence do not raise any resistance). Then, we will explain why XP *practices*, which are in fact those that contribute to the high ranking of XP on the tightness scale (Section 4.2), cause Israeli software teams to express resistance towards XP as soon as they hear the details of the XP practices.

The Israeli culture is open to XP values: In general, Israelis naturally accept the XP values of communication, courage and feedback, mainly due to their military background¹². Israeli software teams, for example, have no problem applying the cooperation that is required by agile methods. Thus, not surprisingly, the majority of those who answered the following question (see Appendix 2) agreed with it: "Teamwork is one ingredient of software development methods". Feedback and the ability to adjust to changes also pose no problem for Israeli software teams. Thus, for example, in one of the interviews, a software developer described his perspective towards software development strategy using the metaphor of war, saying: "If the target is moved, the weapon is changed".

The fact that the Israeli culture is open to the XP values can also partially explain the fact that the CMM model is, to some degree, accepted in Israel. In these two cases, XP values and the CMM model, principles are outlined and a high degree of freedom with respect to their implementation is given.

The Israeli culture resists acceptance of XP practices: It seems like the Israeli software culture does not tend to adopt the XP practices. We explain this fact using our model. Since the Israeli culture is ranked low on the tightness scale, outlining the twelve XP practices, which lead to the high ranking of XP on the tightness scale, gives Israeli software teams a feeling of a highly ordered process that requires on-going examination of one's actions and prevents them from progressing at the speed at which they wish to progress. Furthermore, it should be noted, for example, that some of the XP practices (e.g., refactoring and test-driven development) contradict the improvisational nature of the Israeli software industry mentioned above.

In what follows, we present two examples of data that illustrate clashes between XP and the Israeli culture.

Data 3: Clashes between XP and the Israeli software culture are well illustrated by the resistance raised by the pair programming practice due to the limited freedom it allows.

In our presentation of XP to software engineers in the Israeli hi-tech industry, we use a game (see Appendix 3) in which we rank participants' questions, and answer the questions according to the scores they

¹² Army service is mandatory in Israel.

receive. Following are several questions that were ranked highest by practitioners in our presentations.

- How do we handle constraints that one of the pair members has during the workday (meetings, studies)?
- How do we handle situations in which one of the developers, or both, has other commitments to perform during the day?
- In this method, isn't too much time spent on nice-to-have things, like having two programmers work on a single computer?

We speculate that these questions result from thoughts such as: "Pair programming may have benefits, and it a really nice thing to have in an ideal world. In practice, it is not productive and we can progress faster without it". Deeper analysis might reveal some resistance to the fact that the developer's behavior is limited by the pair mate at his or her side.

Data 4: At the final reflection session of the 2003 Summer Semester, after spending the semester developing a project using XP, the students were asked to indicate the most significant advantage, as well as main disadvantage of XP, as they perceive them to be. Table 6 presents a summary of 41 student reflections, some of which mention more than one item. As can be seen, while issues related to teamwork coordination and to time management are considered to be advantages of XP, in students' responses they are presented equally, both as advantages and disadvantages of XP.

Table 6: Advantages and disadvantages of XP according to undergraduate software engineering students

Topics raised related to	Mentioned as greatest advantage	Mentioned as greatest disadvantage
Teamwork, coordination among team members and work sharing	36	34
Time management, waste of time, investment in planning	13	14

We end this section by explaining the data presented in the Introduction, regarding the high number of start-ups that were established in Israel during the dot.com era. Indeed, in light of what has been described in this article, this fact is not surprising: Low tightness is appropriate for start-ups. In other words, the culture of the Israeli hi-tech industry is suitable for start-ups. At the same time, as Moore indicates (2000), XP is not appropriate for start-ups (cf. Section 2.1). However, since we are passed the dot.com era, we suggest that in order to survive in this post dot.com era, the Israeli hi-tech culture must change its software development culture.

Indeed, this is not a simple process. As it turns out, the Israelis acknowledge the problematic nature of

introducing an SDM into Israeli organizations and are aware of the fact that assimilation of an SDM into their teams is not an easy process. For example, Question 5 of the questionnaire presented in Appendix 2 requests the respondent to indicate YES or NO, according to his or her agreement with several statements. Following are two statements that are relevant to our discussion along with the responses we received (the setting is the same as in Data 1).

- "It is easy to assimilate a software development method in my organization": 74% replied "No", 4% replied "Yes", and 22% did not answer;
- "It is easy to assimilate a software development method in an Israeli software team": 67% replied "No", 11% replied "Yes", and 22% did not answer.

5. Discussion

Ramos, Berry and Carvalho's paper (2002) on *The Role of Emotion, Values, and Beliefs in the Construction of Innovative Work Realities* shows how organizational and national culture impact the implementation and acceptance of Enterprise Resource Planning (ERP) packages (p. 29). In a footnote, it is said of software developers that "Even we software engineers commonly use this assumed implication as an argument against imposing or following a standards-complying mature software development process!" (p. 12).

Our article examined the connections between the national culture and the nature of software development methods. Specifically, we addressed connections between the tightness of the Israeli culture and the tightness of XP. Focus was placed on Israel, a country with a significant influence on the goings-on in the software industry worldwide. Future research, which will be conducted in collaboration with colleagues from other cultures, will explore the other three quarters of Table 1.

It seems that examining the suitability of a culture to an SDM on the national level is more complicated than the more common examination of XP on the organizational level. We suggest expanding our work to the examination of the fitness of XP to different national cultures. In what follows, we present two examples that illustrate some connections between national-cultural issues and software culture.

- o Russia: Edward Yourdon's book entitled "Death march: the complete software developer's guide to surviving 'mission impossible' " was translated to Russian as "The Kamikaze Way".
- o Italy: Francesco Cirillo, one of the leading proponents of the Italian XP community, said in a discussion of the current situation in Italy: "Education of an Extreme Programmer: [...] cultural resistance, difficulties in terms of approach, and future prospects."

We end the paper by asking how suitability between cultures and SDMs is reflected in countries

like India and China, which have entered the software industry only fairly recently. It might not come as a surprise, however, that the first company to attain the highest CMM level (5) was an Indian company.

6. References

Abrahamsson, P., Salo, O., Ronkainen, S. and Warsta, J. (2002). Agile Software Development Methods, Review and Analysis, *ESPOO 2002*, <http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf>

Auer, K and Miller, R. (2001A). *Extreme Programming: Taming the Resistance*, Addison-Wesley.

Auer, K. and Miller, R. (2001B). *Extreme Programming Applied: Playing to Win*, Addison-Wesley.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 2000.

Brooks, F. P. (April 1987). No silver bullet: Essence and accidents of software engineering, *Computer* **20**(4), pp. 10-19.

Hazzan O. (2002). The reflective practitioner perspective in Software Engineering education, *The Journal of Systems and Software* **63**(3), pp. 161-171.

Hazzan, O. and Dubinsky, Y. (2003). Bridging cognitive and social chasms in software development using Extreme Programming, *Proceedings of the Fourth International Conference on eXtreme Programming and Agile Processes in Software Engineering*, pp. 47-53.

Highsmith, J. (2002). *Agile Software developments Ecosystems*, Addison-Wesley.

Kuhn, S. (1998). The software design studio: An exploration, *IEEE software* **15**(2), pp. 65-71.

Kuhn, S. T. (1962). *The Structure of Scientific Revolutions*, University of Chicago Press.

Moore, G. A. (2000). *Living on the Fault Line: Managing for Shareholder Value in the Age of the Internet*. New York: HarperBusiness, 2000.

Mullet, D. (July, 1999). The Software Crisis, *Benchmarks Online - a monthly publication of Academic Computing Services* **2**(7).

Sawyer S. and Guinan, P.J. (1998). Software development: Processes and performance, *IBM Systems Journal* **37**(4), <http://www.research.ibm.com/journal/sj/374/sawyer.html>

Tomayko J. E. (1996). Carnegie-Mellon's software development Studio: A five-year retrospective, *SEI Conf. on Software Engineering Education*.

Tomayko. J. and Hazzan, O. (2004). *Human Aspects of Software Engineering*, Charles River Media.

Walenta, T. (2004). Managing cross-cultural issues in global software outsourcing, *Communications of the ACM* **47**(4), pp. 62-66.

Yourdon, E. (1997). *Death March: The complete software developer's guide to surviving "mission impossible" projects*, N.J.: Prentice Hall PTR.

Ramos, I., Berry, D.M. and Carvalho, J.A. (2002). The Role of Emotion, Values, and Beliefs in *The Construction of Innovative Work Realities*, pp. 300-314.

Appendix 1 - Software Development Questionnaire

For each of the following statements, please mark + in the appropriate column according to your agreement with the statement:

Statement	Agree	Tend to agree	Tend to disagree	Dis - agree
It is important to me that development tasks are allocated equally among team members.				
It is preferable to develop software by planning at the design level, not at the development task level.				
It is accepted in the software industry that developers who frequently go home at 5 PM do not invest enough effort at work.				
It is preferable to minimize, as much as possible, the dependency level among software team members.				
Customers expect that software development will not be completed on time, thus, it is reasonable to commit to an unreasonable timetable.				
The Israeli hi-tech industry is characterized by unplanned software development.				
It is preferable to work in small teams in order to foster decision-making processes as much as possible.				
Single-release software development is preferable over a gradual development process consisting of several releases.				
It is important to enable software developers to work flexible hours ("come when you want, leave when you want").				
If a software project does not proceed as planned, the team must work nights and weekends to catch up.				
No one in my team cares about how the software is written as long as it works.				
A team should extend the development period if it ensures improvement of software quality.				
When I chose a profession, I took into consideration that I would have to devote many hours everyday to work and give up my personal life.				
It is better not to estimate development periods a-priori since software development is characterized by unexpected problems.				
During software development, it is preferable to invest in code readability in order to help future developers whose job will be to maintain the software.				
If software development does not proceed according to the planned schedule, the schedule should be re-planned.				
Intuition and improvisation are important in software development processes.				
It is not important to integrate a reflective process (analysis of the past and learning of lessons) into the software development process.				
When it is difficult to check software, it is OK to move forward and not to insist on testing.				
There is a tendency in my team not to take personal responsibility.				
My team tends to adhere to the timetable.				
Even if I see an opportunity to shorten the development period by skipping tests, I will not take it.				
My team tends to build tight timetables (sometimes by compromising the software quality).				

**Appendix 2 - SwSTE '03 (IEEE International Conference on software – Science, Technology & Engineering)
Questionnaire, Herzelia, Israel, November 2003**

Hello,

The following anonymous questionnaire aims to help us with our research on the teaching of software development methods. Your cooperation is appreciated.

1. In your opinion, what is the major problem of the software industry?
2. Did you learn any software development methods during your undergraduate computer science or software engineering education?
 - Yes, the following method(s): _____
 - No
3. What software development methods have you used so far in your professional career?
4. If you are familiar with an Israeli software team, please indicate what main characteristic of the Israeli software industry led, in your opinion, to the success of the Israeli software industry in the 1990s? _____
5. Please indicate YES or NO, according to your agreement with the following statements.

	YES	NO
Teamwork is one ingredient of software development methods		
Developers should be involved in the definition of requirements		
Customers should define all of their software requirements a-priori		
Testing should be begun only after the development is completed		
It is important to study testing in undergraduate study programs		
Developers should not perform testing		
It is easy to assimilate a software development method in my organization		
It is easy to assimilate a software development method in an Israeli software team		

In case you wish to further contribute to our research from your experience, please attach a business card or complete:

Name: _____ Email: _____

Appendix 3 - Musical Cards

This method was introduced to us at the XP/Agile Universe 2003 conference in Reno by Grigori Melnik. Mr. Melnik used this game in order to reveal the most interesting questions on the topic of testing among an audience of about 70 people. The game took place at the end of a presentation and consisted of the following stages:

1. The participants each received a card (like those used in the planning game) and were asked to write down the question that was most interesting to them.
2. Then the participants were asked to leave their chairs and start walking around.
3. The participants circulated for about thirty seconds, switching cards with whomever they encountered. During these thirty seconds about 10 switches took place. At the end of the thirty seconds, each participant had a card with a question on it, written by someone else.
4. The participants were asked to rank (on the back side of the card) the question that appeared on the card they were holding, according to their interest in the specific question. The ranking

ranged from 5 to 10. This stage took about 20 seconds.

5. Stages 3 & 4 were repeated three times.
6. At this stage, each participant held a card with one question on one side and three ranks on the other side. Each participant was asked to add up the three scores.
7. The session organizer now started to answer the questions in descending order, starting with the questions that received the highest score, 30 points.

In this way, the most interesting questions for at least three of the participants were posed. In the traditional way, questions are posed by people who dare to ask. In addition, the question is formulated in a concise manner, without the accompanying "lecture" that some people tend to add to their questions. Furthermore, this little game also serves as a short, 5-minutes break.

We use this game as a research tool to raise the participants' most important questions. Analysis of these questions can reveal the issues that most bother or interest a specific audience, and shed light on cultural issues in general.